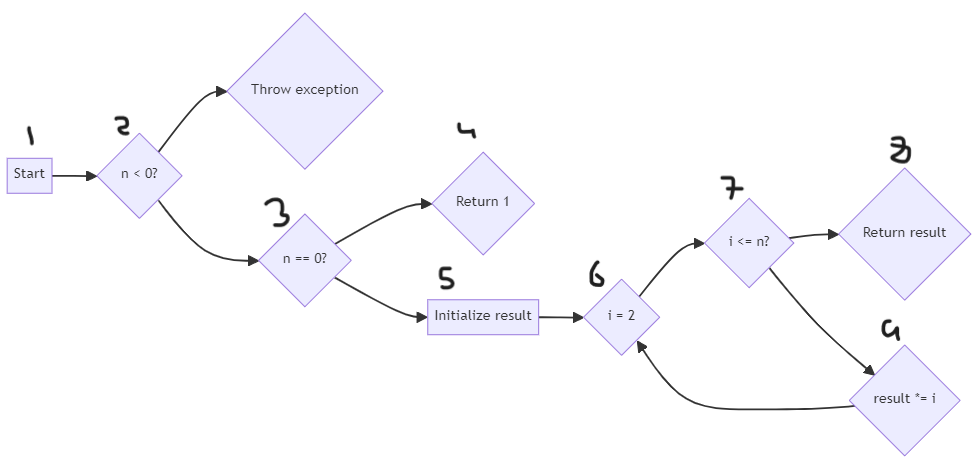
**Code:**

|  |
| --- |
| public class Factorial {    public static long calculateFactorial(int n){  if(n < 0){  throw new IllegalArgumentException("Factorial is not defined");  }  if(n == 0){  return 1;  }    long result = 1;  for(int i=2; i<=n; i++){  result\*=i;  }  return result;  }    public static void main(String[] args){  try{  int num = 5;  long factorial= calculateFactorial(num);  System.out.println("The Factorial of" + num + "is" + factorial );  }catch(IllegalArgumentException e){  System.err.println(e.getMessage());  }  }  } |

**CFG Diagram:**

****

**Paths:**

**Path1:**  1, 2, 3, 4, 5, 6, 7, 8

**Path2:**  1, 2, 3, 4, 5, 6, 7, 9, 6, 7, 8, 9

**Test Cases:**

**Test Cases for Path 1 (n = 0)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Description (Simple English) | Input | Expected Result (Simple English) | Actual Outcome (Pass/Fail) | Coverage Criteria |
| TC-1 | Check factorial for zero | 0 | Should return 1 (base case) | Pass | Statement Coverage, Decision Coverage |

**Table 2: Test Cases for Path 2 (n is negative)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Description (Simple English) | Input | Expected Result (Simple English) | Actual Outcome (Pass/Fail) | Coverage Criteria |
| TC-2 | Try calculating factorial for a negative number | -3 | Should throw an error message | Pass | Statement Coverage, Decision Coverage |